REPORT R-407 FEBRUARY, 1969

CSL COORDINATED SCIENCE LABORATORY

ISL-A STRING MANIPULATING LANGUAGE

K.C. KELLEY
S.R. RAY
F.A. STAHL

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

This work was supported in whole by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB 07-67-C-0199.

Distribution of this report is unlimited.

ISL - A STRING MANIPULATING LANGUAGE

K. C. Kelley
S. R. Ray
F. A. Stahl

Coordinated Science Laboratory
University of Illinois, Urbana, Illinois

ISL - A STRING MANIPULATING LANGUAGE

K. C. Kelley
S. R. Ray
F. A. Stahl

Coordinated Science Laboratory
University of Illinois, Urbana, Illinois

ABSTRACT

In this paper the Information Search Language (ISL) is described. ISL is a problem-oriented language designed to facilitate the manipulation of real character strings.

Introduction

The Information Search Language (ISL) is a problem-oriented language designed to facilitate the manipulation of real character strings with the Control Data 1604 computer. A preliminary form of the language has been described in an earlier report [1].

ISL has demonstrated its usefulness as a string-oriented language in the development of a number of computer usages [2][3][4].

This latest version is an extension and evolution of the previous form. Since the language is currently used only at C.S.L. we are still able to make improvements to it. We have maintained rather successfully its compatibility with its earlier versions.

In Section 1 the ISL instructions are described. The requirements imposed by the ILLAR system are in Section 2. Features of the total system are discussed in Section 3. Sections 4 and 5 survey a number of the existing library routines, program driven and typewriter driven, respectively.

1. Description of ISL Instructions

The ISL language is imbedded in the ILLAR assembly language used at the Coordinated Science Laboratory. Input programs are in card images from paper tape or magnetic tape. The card format is as follows:

```
1    8 10   15 17    20              41
LOC    OPER    B     M-TERM          REMARKS
```

For "pure" ISL instructions the B field is not used. Normally programs are written on the flexowriter and a tab serves to advance the field.

The ISL instructions may be classified as Pseudo-ops, Word-Oriented instructions, Character-string instructions, Transfers, and Input-output instructions.

1.1 Pseudo-ops

1.1.1 EQU - Equate

Form: alpha equ _ x

X is a decimal integer, alpha is the name of a variable. The value of alpha is set equal to x. Note: This instruction is NOT to be included in any execution sequence.

1.1.2 DEFINE - Define variables

Form: ---- define _ x1,x2,x3,x4,x5,...[1]

x1,x2,... are the variables which are to be defined, i.e., memory locations will be reserved for each of these variables. The number of variables which can be defined by this instruction is limited only by the number

---

[1] In describing the form of the instructions, a dotted line in a field will indicate its use is optional, an underscore will indicate it is not to be used.

of them that can be typed on one line. Note: This instruction is NOT to be included in any execution sequence. Also, the value of variables so defined will initially be equal to the (non-zero) core constant.

1.1.3 PAUSE - Pause and proceed.

Form: ---- pause _ x1

Where x1 is the name of a variable. Upon execution of this instruction "pause n" will be typed on the console typewriter, where n is the integer value of the variable x1 and the machine will halt. Restarting the machine starts execution of the next instruction.

1.1.4 STRING - Define a string of characters.

Form: a) ---- string _ ($x_1x_2x_3$... ;), a1,a2

b) ---- string _ alpha, a1, a2

alpha bcd _ (bcd string)
(or)
alpha oct _ (octal number)

The purpose of the string instruction is to assign the variables a1,a2 to the beginning and end addresses respectively of the string of characters $x_1x_2x_3$... . In form a) there may be up to fifty-five characters in the string. In form b) alpha is the address of a string of characters elsewhere in the program that may be entered as either bcd or oct words. In both forms there must be a semi-colon ";" or equivalently a 52_8 following the last character of the string. In form b) the arbitrary limit on the number of characters is eighty.

1.1.5 BEGIN - Define the entry point to an ISL program or subprogram.

Form: _____ begin _ program

The name "program" is assigned as an entry point to the routine. This instruction saves the index registers and makes argument address substitutions if the IDENT card has arguments given. (See Section 2.1.)

1.1.6 RETURN - Define the exit from an ISL program or sub-program.

Form: _____ return _ program

This instruction resets the index registers and exits the routine.

Note:
```
begin    program,x1,x2,x3,...
  :
return   program
```
must be used in pairs.

1.2 Word-oriented Instructions

Since the ISL instructions are all sub-routines, and the value of the accumulator in the CDC 1604 is modified by the sub-routines (e.g., arguments are conveyed by use of the accumulator), a special location is designated as the ISL accumulator (ISLACC). The ISL arithmetic operations are performed on this special accumulator.

1.2.1 LOAD

Form: ---- load _ x1

The value of x1 is placed in ISLACC.

1.2.2 STORE

Form: ---- store _ x1

The value of ISLACC is placed in x1.

1.2.3 PLUS

Form: ---- plus _ x1

The value of x1 is added to ISLACC (fixed point integers only).

1.2.4 MINUS

Form: ---- minus _ x1

The value of x1 is subtracted from ISLACC.

1.2.5 CONVERT - Convert a fixed point number for printing.

Form: ---- convert _ a1,a2

The value of the number in ISLACC is replaced by a string of BCD characters representing the decimal value of ISLACC. A1 and a2 are set to the beginning and end addresses of the string respectively. The converted string must then be removed from the ISLACC before using any other word-oriented instruction.

1.3 Character-oriented instructions

1.3.1 MOVE - Move a string of characters.

Form: ---- move _ a1,a2,b1,b2

The string of characters beginning at the value of a1 and ending at the value of a2 is moved so as to begin at the value of b1 and end at the value of (a2-a1) + b1. I.e., the new beginning (b1) plus the number of characters (a2-a1). The value of b2 is then set equal to the value of the new ending. I.e., b2 = (a2-a1) + b1. This instruction's operation may be denoted by [a1,a2] → [b1,b2]. Note: This instruction transfers characters from [a1,a2] starting at the left. Consequently, if b1 lies between a1 and a2, then the portion of the string [a1,a2] between b1 and a2 will be destroyed before transfer.

1.3.2 SEARCH - Search for a specified string of characters.

Form: a) ---- search _ $(x_1x_2x_3...x_n;)$a1,a2,f,b1,b2

b) ---- search _ alpha,a1,a2,f,b1,b2

A continuous character string, the data string, is assumed to start at the value of a1 and end at the value of a2. The search specification string is given by the characters $x_1x_2x_3...x_n$ in form a) or is defined by alpha in form b). The specifications of $x_1x_2x_3...x_n$ and alpha are described in the STRING instruction. An attempt is made to match the specification string on the data string. If the attempt is successful, the value of f is set positive and the beginning and end addresses of the matched portion of the data string are placed in b1 and b2 respectively. If the match is not successful, f is set negative and b1 and b2 are left undefined.

Any one or more of the x_i (except for x_1 and x_n or two adjacent x's) may be the "don't care" character ":" (colon). The presence of this character as x_i indicates we don't care how many or what characters occur in the data string between the match to x_{i-1} and the match to x_{i+1}.

1.3.3 VSEARCH - Search for a variable string of characters.

Form: ---- vsearch _ s1,s2,a1,a2,f,b1,b2

The s1 and s2 here define the beginning and ending address of the specified string which is to be searched for. Again, a1 and a2 delimit the area to be searched over, f is the success-fail flag, and b1 and b2 define the matched area if one is found.

1.3.4 SEEK - Search for the occurrence of a single character.

Form: ---- seek _ = 1hx,a1,a2

Here "x" is the character to be found and a1 and a2 delimit the area of core to be searched. The routine will search forward or backward depending on whether the value of a1 is less than or greater than the value

of a2. Upon successful return the value of the machine accumulator (A register) contains the address (byte address) of the find. If the character is not found the A will be negative.

1.4 Transfer instructions

1.4.1 IF - Conditional jump

Form: ---- if _ a1,adr1,adr2

The value of a1 is tested. If it is greater than or equal to zero, a conditional jump is made to the program address adr1. If a1 is negative, a jump is made to program address adr2.

1.4.2 GOTO - Unconditional jump

Form: ---- goto _ adr

An unconditional jump is made to program address adr.

1.4.3 TJUMP - A typewriter controlled jump

Form: ---- tjump _ list1,list2,number

This instruction allows the user to pick any one of a number of paths through the execution sequence. Upon execution the typewriter is activated and a tone is emitted to indicate to the user that he is at a branch point in the program which requires that he type a word to indicate his choice of execution path. The lists form two vectors, list1 contains words that he can type, list2 contains the corresponding addresses to which the program execution will jump.

The lists must be of the following form:

```
list1   bcd  _    *word1
        bcd  _    *word2
        bcd  _    *word3
        bcd  :    *wordn
list2   data _    place1
        data _    place2
        data _    place3
        data :    placen
```

The words word1, word2,...,wordn are those that will be matched with the word typed on the typewriter, the places place1,place2,...,placen are addresses in the program to which the jump will be made. "number" has a value equal to the number of items in each list. I.e., "number" has the value n. Note: If an error is made in typing the word on the console typewriter, a space followed by a carriage return will erase that word and allow retyping. If an inappropriate word is typed, the typist will be invited to try again.

1.5 Input-output instructions

ISL magnetic tape functions read and write records of indeterminate length. In order to achieve variable length read the first word of a magnetic tape record is a number giving the number of words (multiples of eight characters) that are on the record and the byte position (0 through 7) of the last meaningful character. Input data can easily be put into this form by use of an auxiliary program IBMISL, which translates BCD records of

fixed length into prefixed records readable by ISL. The ISL write instruction takes care of this bookkeeping itself so that tapes written with ISL write can be read with no problem.

1.5.1 TSTRING - Input a string from the typewriter

Form: ---- tstring _ a1,a2

When this instruction is encountered at execution time, the typewriter is activated and a tone is emitted. A string may then be typed on the typewriter followed by a carriage return. A1 and a2 will be set equal to the beginning and end addresses respectively of the typed string.

Note: Because of the difference of codes transmitted by the typewriter, the don't care character ":" and the end of string character ";" must be typed on the console typewriter as "$" and "%" respectively.

1.5.2 READ - Read from magnetic tape

Form: ---- read _ tape,nd,ef

The next record of magnetic tape is read from the unit whose logical number is the value of tape. The record is read into core beginning at location MID (see predefined variables). The end of the record read in is calculated and stored at nd. In the event an end of file is reached, the message "EOF" is printed on the console typewriter and a jump is made to the program address ef. Each record is checked for length, parity, and buffer errors before the next instruction is executed.

1.5.3 WRITE - Write a record on magnetic tape

Form: ---- wirte _ tape, loc

A record is written in standard ISL format (binary, variable length, and with prefix word) on the tape whose logical number is the value of tape. The character string to be written begins at the value of ANT

(see predefined variables) and ends at the value of loc. Records written are checked before the next instruction is executed.

1.5.4 EFMARK - Write an end of file mark on magnetic tape.

Form: a) ---- efmark _ tape

b) ---- wrteof _ tape

An end of file mark (six inches of blank tape followed by the number 1717) is written on the magnetic tape whose logical number is the value of tape. The value must be an integer from one to eight.

1.5.5 REWIND - Rewind a magnetic tape

Form: ---- rewind _ tape, mode

The magnetic tape whose logical number is the value of tape is rewound to the load point. Mode indicates whether it is to be a read rewind or a write rewind. If mode is negative it is a read rewind; if mode is positive it is a write rewind.

1.5.6 BACK - Backspace the tape one record.

Form: ---- back _ tape,mode

The tape whose logical number is the value of tape is backspaced one record. If the value of mode is negative it backspaces in position for the read head; if the value of mode is positive it backspaces for the write head.

1.5.7 PRINT - Print on the line printer.

Form: ---- print _ beg,nd,col,cc

The character string beginning at the value of the variable beg and ending at the value of nd is printed on the line printer. The value of "col" is the column in which the initial line will start. Before

printing the page is moved upward according to the value of "cc".

value of cc =
0 - move upward single space
1 - move to top of next page
2 - move upward z spaces
3 - suppress spacing after printing

The length of the character string to be printed has no restriction. If it will not fit on one line (112 spaces) the remaining portion is printed on succeeding lines.

Display Output

The following instructions allow the user to display strings of characters on the display screen directly from their core location with a minimum of programming effort. While the display is on in this fashion the two words immediately preceding the string in core are replaced by control words and the area is used as a buffer. When the display is turned off these words in core are restored. There are two entry and exit points in this routine, either can be used according to one's needs.
Note: The programmer is expected to have his own carriage returns in the string of characters. (A subroutine to do this is available - see section 4.)

1.5.8 ISLTV - Display a string of characters.

Form: ---- isltv a1,a2

The string of characters from value a1 to value a2 is displayed. No checking is done to see if sufficient carriage returns are present or whether too many lines (more than 48) are being displayed. The assumption here is that the user knows his string is short enough to fit comfortably on the screen. The display stays on and the program waits, until control

is returned by the operator pushing carriage return on the typewriter. If this method is used TVOFF is not needed. The user familiar with ILLAR will have another option, that of immediate return to calling program, with the scope on. He must then turn it off later by TVOFF.

1.5.9 TVOFF/STOPTV - Cease display of character string.

Form: a) ---- tvoff _ _____

b) ---- stoptv_ _____

The circulating buffer for the scope is turned off, the areas used by the control words and leading and trailing characters are restored.

1.5.10 STRTTV - Start the display with control.

Form: ---- strttv _ a1,a2

The characters between value of a1 and value of a2 are displayed. However, the display unit has the property that any attempt to put more than about 1000 characters of this size on at once will cause a flickering of the screen. Also in the event too many lines are required the image will "wrap-around" bottom to top. In order to circumvent this problem this entry point initiates a series of checks. The number of characters and carriage returns is automatically held to a number which will display nicely on the screen. This portion of the string is then displayed and four symbols (↑,↓,P,→) are displayed on the bottom of the screen. Pressing the light pen against the "long-line" portion of any of the symbols will cause appropriate changes or actions in the display. "↑" will cause the images to "move-up", pushing lines off the top and putting as many as possible in at the bottom. "↓" performs the same function in the other direction. In the event that the beginning or end of the character string

is on the screen so that movement in that direction is impossible, the appropriate "arrow-shaft" will go out. Pressing the light pen against "→" will return control to the calling program. Pressing "P" will cause a photo to be taken of the characters on the screen (without the special symbols). As an added convenience, at any time that the display is on with the symbols along the bottom the operator may type: "carriage return" followed by the word "print" followed by "carriage return" and the entire scope buffer (character string) will be printed on the line printer in the same format as on the screen.

It must be remembered that the programmer must be sure his carriage returns (32 octal) are in the string if it exceeds 96 characters. For this entry, designed for long strings, the absence of any carriage returns is taken to be an error. An appropriate message is printed and the machine stops. Upon restarting the program will put in carriage returns in place of blanks at approximately every 96 characters and display it anyway.

1.6 Predefined Symbols

The following variable names are predefined in all ISL programs. The values placed in these locations are calculated and set upon execution of the first "BEGIN" statement after loading.

1.6.1 ANT - The value of ANT is the first location of the character organized portion of core. The "WRITE" instruction always takes the value of ANT to be the location of the first character to be written.

1.6.2 MID - The value of MID is the "middle" of the character organized portion of core. The "READ" statement brings character strings

into core beginning at the location given by the value of "MID".

1.6.3 POST - The end of character organized core. The length of a record to be read is always checked to be sure the record will not overlap this location and wipe out the program in "wrapping-around" core on read-in.

1.6.4 NXT - This is designed for convenience only. It is often used, at the discretion of the programmer, to hold the location of the end of a string read-in.

1.6.5 ISLACC - The ISL accumulator. All ISL arithmetic functions, as well as "CONVERT" are performed on this word of core. The symbol ISLACC can be used as the first argument of an "IF" statement.

2. Requirements Imposed by the ILLAR System

2.1 Ident card - Each ISL program must have as its very first "card" an identification card.

Form: _____ ident _ name,---,---

This defines the name of the program, which may or may not be the same name given by the entry statement BEGIN. "Name" may be followed by arguments for the program or sub-program. If there are arguments they must be used in the program with the same names as on the IDENT "card". When the program is called by a "CALL" statement (see section 3), the BEGIN statement automatically takes care of transferring argument values.

2.2 Lib ISL Card - Each ISL program must have as its second card the following:

Form: _____ lib _ isl

The function of this pseudo-op in the ILLAR system is to generate the series of macro definitions and variable declarations required by all ISL programs. These are not normally seen in the program listing.

2.3 END Card - Each program must have as its last "card" an END card.

Form: ---- end _ _____

2.4 FINIS Card - Each group of programs (one or more) must have following the last END card a FINIS card.

Form: _____ finis _ _____

3. Features of the Total System

Since the ISL language is imbedded in the ILLAR system, all of the features of the ILLAR system are immediately available to the ISL programmer. Some familiarity with this system is to be desired for the ISL programmer if he is to make use of the full range of capabilities of the system. In the meantime, or for the new programmer, some of the features have been found to be quite useful for writing ISL programs with a minimum of difficulty. All of these features are described more fully in the ILLAR write-ups.

Any machine language instruction may be used in an ISL program. In fact, the user whose previous experience with the 1604 is in machine language or the ISL programmer who has written several programs will find that he tends to use machine language most of the time, using the ISL instructions only as he needs them. He can then use the index registers and their associated instructions, use the equality search instructions for certain kinds of table lookups, write special input-output subroutines or do whatever he can do in the normal machine language.

The ILLAR system allows the use of literals so that the programmer does not have to define variables for every number he wishes to use in the M-field. For example, he may write:

```
----      read _  =8,nxt,ef
```

and not have to worry about defining the number 8 somewhere else in his program. In a print instruction the programmer may write, for instance:

```
----      print _  mid,nxt,=1,-0
```

to print a string starting in column one and single spaced.

Macros may be used to generate instruction "cards" during the

assembly process. For example, one may define a macro BIT18 to form an 18-bit byte address from the 15-bit address of a beginning of a string or a table. The macro definition would look like:

```
BIT18    MACRO    A
         ENA      A
         SCL      =-77777b
         ALS      3
         ENDM
```

This definition would appear at the beginning of the program. Then when the process is needed, say to find the byte address of a string defined with a BCD statement labelled STR, one would write:

```
-----    BIT18    STR
```

and the assembler would generate the cards needed to expand the macro:

```
         ENA      STR
         SCL      =-77777b
         ALS      3
```

There are several ways in which the programmer can write and call upon subroutines and subprograms. The simplest of these is the subroutine that is called without arguments and which is written within the program itself (i.e., the subroutine lies between the same IDENT and END cards that the using program does.) Such a subroutine could be used as follows:

```
         IDENT    MAINPROG
         LIB      ISL
         BEGIN    MAINPROG
...      ...    . ...
...      ...    . ...
         SUBPR
...      ...    .. ...
...      ...    . ...
         RETURN   MAINPROG
SUBPR    ENTER
...      ...    . ...
...      ...    . ...
         EXIT
         END
         FINIS
```

It should be noted here that the BEGIN and RETURN statements are ISL instructions and can be used only if the LIB ISL card is in the beginning of the program. In a normal ILLAR program these would be replaced by the cards:

```
MAINPROG  ENTRY
```

and

```
EXIT      MAINPROG
```

respectively.

Now in the case that SUBPR is to be a completely separate program with its own IDENT and END cards its entry must be the same as any other main program, for example, like MAINPROG above. However, when it is then to be called from MAINPROG, the card

```
SUBPR
```

above must be replaced by a CALL to the subroutine:

```
CALL      SUBPR
```

In both of these uses the subroutine or subprogram had no arguments. ILLAR provides some very nice tools for subroutine linkages and for the ISL programmer, these tools are automatically used by writing a BEGIN statement and a RETURN statement. The calling program and the subprogram would be of a form similar to the following:

```
        IDENT     MAINPROG
        LIB       ISL
        BEGIN     MAINPROG
...     ...     . ....
...     ...     . ....
        CALL      SUBPR,VAR1,VAR2
...     ...     . ....
...     ...     . ....
        END

        IDENT     SUBPR,ARG1,ARG2
        LIB       ISL
        BEGIN     SUBPR
...     ...     . ....
...     ...     . ....
...     ...     . ....
        RETURN    SUBPR
        END
        FINIS
```

The body of the subprogram operates on the arguments.

Other variations on the use of CALL, PREAMB, SAVEB AND RESETB, EXT, ENTRY, and EXIT are available to the programmer familiar with ILLAR.

Since the ISL instructions are actually subprograms in themselves, they are available for use as subroutines in FORTRAN programs. In this case they are used by writing a standard FORTRAN CALL statement, with the name of the arguments following the name of the routine. In a very few cases the name to be used in the call statement is different from the actual ISL instruction. Briefly, these are the instructions in ISL which have the same name as standard FORTRAN subroutines, e.g., PRINT, IF, etc.

4. Useful Program Driven Subroutines

The following subroutines, while not in the exact form of ISL instructions, have been written as the need arose in the process of writing imbedded systems.[2][3] The ISL programmer may find some of them to be useful in writing ISL programs. The first three are so useful that it is anticipated that at some future time they will become part of the set of "pure" ISL instructions.

4.1 POP - Pop one character out of core.

FORM: CALL POP

Upon entry the 18-bit address of the character to be popped is in the Q. On exit the character is in the A right justified with zeros left. The updated pointer is in Q.

4.2 PUSH - Push one character into core

FORM: CALL PUSH

Upon entry the octal code of the character to be pushed is in index register 2 right justified with zeros left. The 18-bit address of the byte to be filled is in the A upon entry. There are no exit parameters.

4.3 LJUMP - Jump as directed by the light pen.

FORM: CALL LJUMP,LIST1,LIST2,LNGTH

This subroutine is the light-pen analogy to the TJUMP instruction. LIST1 is a list of words that will be displayed on the screen with a pointer next to each of them. LIST2 is a list of program addresses to which the corresponding jumpwill be made when the light pen is pressed against the appropriate pointer on the screen. LNGTH gives the length of each of the lists; currently the length must not exceed 20. In the event that the

light pen is not working, this instruction becomes a TJUMP simply by pressing carriage return on the console typewriter.

4.4 FIXLINE - Insert carriage returns in a line to be displayed.

FORM: a) CALL FIXLINE,AA,BB

b) CALL CRLINE,AA,BB

The string of characters beginning at the value of AA and ending at the value of BB is divided into segments of less than 96 characters so that the line will not "wrap around" when it is displayed on the scope. The divisions are made by inserting the octal code for a carriage return in place of the space before the word that would overlap. The CRLINE entry insures that a carriage return will be put at the end of the entire string.

4.5 FXAREA - Insert leading and trailing blanks.

FORM: CALL FXAREA,AA,BB

This routine fills out remaining portions of the beginning and ending words of the string starting at the value of AA and ending at the value of BB. The beginning word is filled out with blanks left, the ending word with blanks right. The words in core that are destroyed in this process are saved internally and may be restored by using the entry UNFXAR:

FORM: CALL UNFXAR

4.6 PREPTV - Prepare a series of lines of characters separated by minuses for display.

FORM: CALL PREPTV,AA,BB

The string of characters beginning at the value of AA and ending at the value of BB is searched for all occurrences of a minus (40 octal) and this character is replaced by a carriage return (32 octal). In the event

that the distance between two minuses is 95 or greater, carriage returns are inserted as needed by calling CRLINE.

4.7 PRNTSCOP - Print out the information in the scope buffer.

FORM: CALL PRNTSCOP,AA,BB

The string of characters beginning at the value of AA and ending at the value of BB is assumed to be in the correct form on the scope. The lines are printed on the line printer exactly as they would occur on the screen, i.e., carriage returns (32 octal) separate the lines.

5. Typewriter Driven Routines

The following routines are normally driven from the console typewriter.[1] If an entry point exists to drive the routine from another program, that entry point will be mentioned. For each of the routines there is a standard set of arguments which will be used if no arguments are given or if any argument is left out by typing two commas with no argument in between.

5.1 SYNTAX - A routine to check syntax on the data.

FORM: go,syntax,n

Here n is an integer actually typed on the typewriter. It must have a value between one and eight and indicates which tape is to be checked.

This routine checks this ISL data and prints appropriate messages along with the offending record on the line printer. It does not catch all possible errors, but it does find misplaced cards, missing continuation numbers, multiple titles, and a variety of assorted typing errors that affect syntax. The standard argument for n is 2.

5.2 IBMISL - Translate BCD records of fixed length to variable length form.

FORM: go,ibmisl,n,m,l

Here n,m,l are integers designating the input tape, output tape, and the length of the records on the input tape; if arguments are not provided these are 2,3, and 10, respectively. The records on the input tape are assumed to be in BCD mode and all of the same length. The read is done on an interrupt basis and the write keeps up as best it can. When core is read full, the read waits for write to catch up and then they start over at the beginning of available core. WARNING: There must be an end of file on

---

[1] The routines in this section have been written for specific operations in imbedded systems [2][3][4] and are included here because of their general usefulness.

the input tape! There are standard programs available to put IBM cards on a tape to use as input to this routine, but the input tape must be written at a density of 200 bpi to be read on the 1604's tape units!

5.3 LISTISL - List a tape written in ISL records.

FORM: go,listisl,t,h,c

T, h, and c are all integers. T gives the tape number to be listed. H tells how it is to be listed in the following sense: if h is 0, the tape will be listed exactly as it is, but if h is 1, strings of characters between minuses on the record will be printed as lines, that is, the strings will be "unpacked". Standard arguments are 2,0,0 respectively. C is the carriage control indicator, the same as in PRINT.

5.4 COPYISL - Copy ISL records from one magnetic tape to another.

FORM: go,copyisl,n,m

The tape designated by the integer n is copied onto the tape designated by the integer m until the first end of file. The end of file is also copied. N and m may be any integers between 1 and 8. The standard arguments are 2 and 3, respectively.

5.5 PACK - Pack the data from card images to squeezed form.

FORM: go,pack,n,m

The ISL data card images on tape n are packed onto the tape m. In this process, all excess blanks are deleted and the data records between the "("card and the ")" card are packed into one record. The standard arguments are 2 and 3, respectively.

5.6 TITLES - Separate the titles from the data and write them on a separate tape.

FORM: go,titles,n,m

The titles of all source and citation documents on tape n are written out on tape m. A count is printed of the total number of source titles and citation titles. The standard arguments for this routine are 2 and 3, respectively.

5.7 PRESORT - Prepare records for sorting on all words in the string.

FORM: go,presort,n,m

The strings of characters on the input tape n are formed into multiple records such that each word starts in a given column for sorting. The records so composed are each 300 characters long with the sorting column as column 150. This routine is used primarily to get titles ready to sort into a key-word in title listing.

5.8 POSTSORT - Print sorted character strings.

FORM: go,postsort,n

The records, which are assumed to have been sorted, are printed out in a form somewhat like a standard KWIC index to titles. The sorted column lines up and the rest of the record is rotated around the end of the page. Only the 50 columns on either side of the sorting column are printed. The records are assumed to have an identifier. This is printed at the right-hand end of the line.

5.9 SHOW - Show the data on the screen in an unpacked form.

FORM: go, show

This routine has no arguments. The data in packed form is assumed to be on tape unit eight. The ISL records are read off one at a time in sequence and displayed on the screen. When the user desires to go on to another document he simply presses the light pen against the "out" arrow and

the typewriter will type the characters "cs". If the user wishes to see the next document, he types the character c followed by a carriage return; if he wishes to stop, he types the character s followed by a carriage return.

5.10 TSTDSPLY - Display the documents in unpacked form.

FORM: go,tstdsply

This routine has no arguments. The data are always assumed to be on tape 8. The display here is in the packed form essentially as it appears on the tape or in core. In this routine, when the "out" arrow is pushed, there is no typewriter response. The program waits for the user to type either "cont" or "stop" on the typewriter, followed by a carriage return.

5.11 FINDITEM - Find the document with a given item number on the tape.

FORM: go,finditem,n,m

The integer n denotes which tape unit the data is written on, while the integer m denotes the item number of the initial item to be found. When the item is found, it is displayed. On return from the display routine (by pressing the "out" arrow) the user is presented with a choice by way of the light pen. The three keys are "hold", "thanks", and "another". Pointing to "hold" will cause the document to be displayed again; pointing to "another" will initiate a request from the typewriter that the user type the number of the item he now wants; and pointing to "thanks" will exit the routine.

6. Acknowledgment

The authors gratefully acknowledge the aid and valuable suggestions of Mr. Donald Lee in implementing ISL.

Thanks are also due to the entire staff associated with the computer installation at Coordinated Science Laboratory for their many hours of assistance and patience.

REFERENCES

1. K. C. Kelley, S. R. Ray, F. A. Stahl, INFORMATION SEARCH LANGUAGE, File No. 735, Dept. of Computer Science, Univ. of Ill., Urbana, Illinois, September 12, 1967. Supported in part by Contract AT(11-1)-1018 with the U.S. Atomic Energy Commission and the Advanced Research Projects Agency.

2. D. E. Carroll, R. T. Chien, K. C. Kelley, F. P. Preparata, S. R. Ray, P. R. Reynolds, F. A. Stahl, "AN INTERACTIVE DOCUMENT RETRIEVAL SYSTEM," CSL Report R-398, University of Illinois, Urbana, Illinois, December 1968.

3. R. T. Chien, K. C. Kelley, F. A. Stahl, "Implementation of the REQUEST Document Retrieval System," to appear, CSL, Univ. of Illinois, Urbana, Illinois.

4. James Merritt Jansen, Jr., "Phrase Dictionary Construction Methods for the R2 Information Retrieval System," to appear, CSL, University of Illinois, Urbana, Illinois.

5. D. E. Carroll, "Guidelines for INFORMATION RETRIEVAL DATA BASE," to appear, CSL, University of Illinois, Urbana, Illinois.

Distribution List as of November 1, 1968

1 Dr A.A. Dougal
Asst Director (Research)
Ofc of Defense Res & Eng
Department of Defense
Washington, D.C. 20301

1 Office of Deputy Director
(Research and Technology)
ODD R&E-OSD
The Pentagon, Room 3-E-144
Washington, D.C. 20301

1 Director Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Director for Information Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Director for Materials Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Headquarters
Defense Communications Agency (333)
The Pentagon
Washington, D.C. 20305

20 Defense Documentation Center
Attn: TISIA
Cameron Station, Bldg 5
Alexandria, Virginia 22314

1 Director
National Security Agency
Attn: Librarian C-332
Fort George G. Meade, Maryland 20755

1 Weapons Systems Evaluation Group
Attn: Col Daniel W. McElwee
Department of Defense
Washington, D.C. 20305

1 National Security Agency
Attn: R4-James Tippett
Office of Research
Fort George G. Meade, Maryland 20755

1 Central Intelligence Agency
Attn: OCR/DD Publications
Washington, D.C. 20505

1 Colonel Kee
AFRSTE
Hqs, USAF
Room ID-429, The Pentagon
Washington, D.C. 20330

1 Aerospace Medical Division
AMD (AMRXI)
Brooks Air Force Base, Texas 78235

1 AUL3T-9663
Maxwell AFB, Alabama 36112

1 AFFTC (FTBPP-2)
Technical Library
Edwards AFB, Calif. 93523

1 Hq SAMSO (SMITA/Lt Nelson)
AF Unit Post Office
Los Angeles, California 90045

1 Lt Col Charles M. Waespy
Hq USAF (AFRDSD)
Pentagon
Washington, D.C. 20330

1 SSD (SSTRT/Lt Starbuck)
AFUPO
Los Angeles, California 90045

1 Det #6, OAR (LOOAR)
Air Force Post Office
Los Angeles, California 90045

1 ARL (ARIY)
Wright-Patterson AFB, Ohio 45433

1 Dr H.V. Noble
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 Mr Peter Murray
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 AFAL (AVTE/R.D. Larson)
Wright-Patterson AFB, Ohio 45433

2 Commanding General
Attn: STEWS-WS-VT
White Sands Missile Range
New Mexico, 88002

1 RADC (EMLAL01)
Griffiss AFB, New York 13442
Attn: Documents Library

1 Mr H.E. Webb (EMIA)
Rome Air Development Center
Griffiss AFB, New York 13442

1 Academy Library (DFSLB)
U.S. Air Force Academy
Colorado Springs, Colorado 80912

1 Mr Morton M. Pavane, Chief
AFSC Scientific and Liaison Office
26 Federal Plaza
New York, N.Y. 10007

1 Lt Col Bernard S. Morgan
Frank J. Seiler Research Laboratory
U.S. Air Force Academy
Colorado Springs Colorado 80912

1 Technical Library, AFETR
(ETV, MU-135)
Patrick AFB, Florida 32925

1 AFETR (ETLLG-1)
STINFO Office (For Library)
Patrick AFB, Florida 32925

1 Dr L. M. Hollingsworth
AFCRL (CRN)
L.G. Hanscom Field
Bedford, Massachusetts 01731

1 AFCRL (CRMZLR)
AFCRL Research Library, Stop 29
L.G. Hanscom Field
Bedford, Mass 01731

1 Colonel Robert E. Fontana
Dept of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

1 Colonel A.D. Blue
RTD (RTTL)
Bolling Air Force Base, D.C. 20332

1 Dr I.R. Mirman
AFSC (SCT)
Andrews AFB, Maryland 20331

1 AFSC (SCTR)
Andrews AFB, Maryland 20331

1 Lt Col J.L. Reeves
AFSC (SCBB)
Andrews AFB, Maryland 20331

2 ESD (ESTI)
L.G. Hanscom Field
Bedford, Mass 01731

1 AEDC (ARO, INC)
Attn: Library/Documents
Arnold AFS, Tenn 37389

2 European Office of Aerospace Research
Shell Building
47 Rue Cantersteen
Brussels, Belgium

5 Lt Col Robert B. Lalisch
Chief, Electronics Division
Directorate of Engineering Sciences
Air Force Office of Scientific Research
Arlington, Virginia 22209

1 APGC (PGBS-12)
Elgin AFB, Florida 32542

1 U.S. Army Research Office
Attn: Physical Sciences Division
3045 Columbia Pike
Arlington, Virginia 22204

1 Research Plans Office
U.S. Army Research Office
3045 Columbia Pike
Arlington, Virginia 22204

1 Commanding General
U.S. Army Materiel Command
Attn: AMCRD-TP
Washington, D.C. 20315

1 Commanding General
U.S. Army Strategic Communication Command
Fort Huachuca, Arizona 85613

1 Commanding Officer
Army Materials & Mech. Res. Center
Watertown Arsenal
Watertown, Mass. 02172

1 Commanding Officer
U.S. Army Ballistics Research Laboratory
Attn: AMXRD-BAT
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commandant
U.S. Army Air Defense School
Attn: Missile Sciences Division
C & S Dept
P.O. Box 9390
Fort Bliss, Texas 79916

1 Commanding General
U.S. Army Missile Command
Attn: Technical Library
Redstone Arsenal, Alabama 35809

1 U.S. Army Munitions Command
Attn: Technical Information Command
Picatinney Arsenal
Dover, New Jersey 07801

1 Commanding Officer
Harry Diamond Laboratories
Attn: Dr Berthold Altman (AMXDO-TI)
Connecticut Ave & Van Ness St. N.W.
Washington, D.C. 20438

1 Commanding Officer
U.S. Army Security Agency
Arlington Hall
Arlington, Virginia 22212

1 Commanding Officer
U.S. Army Limited War Laboratory
Attn: Technical Director
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commanding Officer
Human Engineering Laboratories
Aberdeen Proving Grounds
Aberdeen, Maryland 21005

1 Commandant
U.S. Army Command & General Staff
College
Attn: Secretary
Fort Leavenworth, Kansas 66270

1 Commanding Officer
U.S. Army Research Office (Durham)
Attn: CRD-AA-IP (Richard O. Ulsh)
Box CM, Duke Station
Durham, North Carolina 27706

1 Librarian
U.S. Army Military Academy
West Point, New York 10996

1 The Walter Reed Institute of Research
Walter Reed Medical Center
Washington, D.C. 20012

1 Commanding Officer
U.S. Army Electronics R & D Activity
White Sands Missile Range
New Mexico 88002

1 Dr H. Robl
Deputy Chief Scientist
U.S. Army Research Office (Durham)
Box CM, Duke Station
Durham, North Carolina 27706

1 U.S. Army Mobility Equipment
Research & Development Center
Attn: Technical Document Center
Bldg 315
Fort Beloir, Virginia 22060

1 Mr Norman J. Field (AMSEL-RD-S)
Chief, Office of Science & Technology
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703

1 Mr Robert O. Parker
Executive Secretary,
JSTAC (AMSEL-XL-D)
Fort Monmouth, New Jersey 07703

1 Commanding General
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703
Attn: AMSEL-SC
RD-D
RD-G
RD-GF
RD-MAT
XL-D
XL-E (Dr K. Schwidta)
XL-C
1 Cy to each symbol listed. XL-S
HL-D
HL-CTR
HL-CT-P (Dr W. McAfee)
HL-CT-L
HL-CT-O
HL-CT-I
HL-CT-A
NL-D
NL-A
NL-P-2 (D. Haratz)
NL-R
NL-S
KL-D
KL-E
KL-S
KL-T
VL-P (R.J. Niemela)
WL-D

1 Director Night Vision Laboratory
U.S. Army Electronics Command
Attn: HL-NV-II (Dr A.D. Schnitzler
Fort Belvoir, Virginia 22060

1 Components Research Laboratory
(P.E. Landis) Bldg 92
Harry Diamond Laboratories
Connecticut Ave & Van Ness St N.W.
Washington, D.C. 20438

1 Mr Edward Vaughan
Research & Engineering Directorate
U.S. Army Weapons Command
Rock Island, Illinois 61201

1 Commanding General
U.S. Army Missile Command
AMSMI-REX (W. Todd)
Redstone Arsenal, Ala 35809

3 Chief of Naval Research
Department of the Navy
Washington, D.C. 20360
Attn: Code 427

2 Naval Electronics Systems Command
ELEX 03
Falls Church, Virginia 22046

1 Naval Ship Systems Command
SHIP 031
Washington, D.C. 20360

1 Naval Ships Systems Command
SHIP 035
Washington, D.C. 20360

2 Naval Ordnance Systems Command
ORD 32
Washington, D.C. 20360

2 Naval Air Systems Command
AIR 03
Washington, D.C. 20360

2 Commanding Officer
Office of Naval Research Branch Office
Box 39, Navy No.100 F.P.O.
New York, New York 09510

1 Commanding Officer
Office of Naval Research Branch Office
219 South Dearborn Stree
Chicago, Illinois 60604

1 Commanding Officer
Office of Naval Reserve Branch Office
207 West 24th Street
New York, New York 10011

1 Commanding Officer
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California 91101

1 Commanding Officer
Office of Naval Research Branch Office
495 Summer Street
Boston, Massachusetts 02210

8 Director, Naval Research Laboratory
Technical Information Officer
Washington, D.C. 20360
Attn: Code 2000

1 Commander
Naval Air Development & Material Center
Johnsville, Pennsylvania 18974

2 Librarian
U.S. Naval Electronics Laboratory
San Diego, California 95152

1 Commanding Officer & Director
U.S. Naval Underwater Sound Laboratory
Fort Trumbull
New London, Connecticut 06840

1 Librarian
U.S. Naval Post Graduate School
Monterey, California 93940

1 Commander
U.S. Naval Air Missile Test Center
Point Magu, California 93041

1 Director
U.S. Naval Observatory
Washington, D.C. 20390

2 Chief of Naval Operations
OP-07
Washington, D.C. 20350

1 Director, U.S. Naval Security Group
Attn: G43
3801 Nebraska Avenue
Washington, D.C. 20390

2 Commanding Officer
Naval Ordnance Laboratory
White Oak, Maryland 21502

1 Commanding Officer
Naval Ordnance Laboratory
Corona, California 91720

1 Commanding Officer
Naval Ordnance Test Station
China Lake, California 93555

1 Commanding Officer
Naval Training Device Center
Orlando, Florida 32811

1 Commanding Officer
Naval Avionics Facility
Indianapolis, Indiana 46241

1 U.S. Naval Weapons Laboratory
Dahlgren, Virginia 22448

1 Weapons Systems Test Division
Naval Air Test Center
Patuxtent River, Maryland 20670
Attn: Library

1 Head, Technical Division
U.S. Naval Counter Intelligence
Support Center
Fairmont Building
4420 North Fairfax Drive
Arlington, Virginia 22203

1 Mr Charles Yost
Special Asst to the Director of Res.
National Aeronautics & Space Admin.
Washington, D.C. 20546

1 Dr H. Harrison, Code RRE
Chief, Electrophysics Branch
National Aeronautics & Space Admin.
Attn: Library C3/TDL
Green Belt, Maryland 20771

1 NASA Lewis Research Center
Attn: Library
21000 Brookpark Road
Cleveland, Ohio 22135

1 National Science Foundation
Attn: Program Director
Engineering System Program ENG
1800 G. Street, N.W.
Washington, D.C. 20550

1 U.S. Atomic Energy Commission
Division of Technical Information Ext.
P.O. Box 62
Oak Ridge, Tenn. 37831

1 Los Alamos Scientific Laboratory
Attn: Reports Library
P.O. Box 1663
Los Alamos, New Mexico 87544

2 NASA Scientific & Technical Inform. Fac.
Attn: Acquisitions Branch (S/AK/DL)
P.O. Box 33
College Park, Maryland 20740

1 Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Mass 02139

1 Polytechnic Institute of Brooklyn
55 Johnson Street
Brooklyn, New York 11201
Attn: Mr Jerome Fox
Research Coordinator

1 Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, New York 10027

1 Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

1 Director
Stanford Electronics Laboratories
Stanford University
Stanford, California 94305

1 Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

1 Director
Electronic Sciences Laboratory
University of Southern California
Los Angeles, California 90007

1 Electronics Research Center
University of Texas at Austin
Engr-Science Bldg 110
Austin, Texas 78712

1 Division of Engineering &
Applied Physics
210 Pierce Hall
Harvard University
Cambridge, Massachusetts 02138

1 Aerospace Corporation
P.O. Box 95085
Los Angeles, California 90045
Attn: Library Acquisition Group

1 Professor Nicholas George
California Inst. of Technology
Pasadena, California 91109

1 Aeronaurics Library
Graduate Aeronautical Laboratories
California Institute of Technology
1201 E. California Blvd
Pasadena, California 91109

1 Director, USAF Project Rand
Via: Air Force Liaison Office
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

1 Hunt Library
Carnegie Institute of Technology
Schenley Park, Pittsburgh, Pa. 15213

1 Syracuse University
Dept of Electrical Engineering
Syracuse, New York 13210

1 Yale University
Engineering Dept
New Haven, Connecticut 06520

1 Airborne Instruments Laboratory
Deerpark, New York 11729

1 Bendix Pacific Division
11600 Sherman Way
North Hollywood, California 91605

1 General Electric Co
Research Laboratories
Schenectady, New York 12301

1 Lockheed Aircraft Corp
P.O. Box 504
Sunnyvale, California 94088

1 Raytheon Co
Bedford, Mass 01730
Attn: Librarian

1 Dr G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

1 Dr John C, Hancock, Director
Electronics Systems Research Laboratory
Purdue University
Lafayette, Indiana 47907

1 Director
Microwave Laboratory
Stanford University
Stanford, California 94305

1 Emil Schafer, Head
Electronics Properties Infor. Center
Hughes Aircraft Co
Culver, California 90230

1 The John Hopkins University
Applied Physics Laboratory
8621 Georgia Avenue
Silver Springs, Maryland 20910
Attn: Boris W. Kuvshinoff
Document Librarian

1 Dr Leo Young
Stanford Research Institute
Menlo Park, California 94025

1 Mr Henry Bachmann
Assistant Chief Engineer
Wheeler Laboratories
122 Cutterhill Road
Great Neck, New York 11021

1 School of Engineering Sciences
Arizona State University
Tempe, Arizona 85281

1 Engineering & Math Sciences Library
University of California at Los Angeles
405 Hilgred Avenue
Los Angeles, California 90024

1 California Institute of Technology
Pasadena, California 91109
Attn: Documents Library

1 University of California
Santa Barbara, California 93106
Attn: Library

1 Carnegie Institute of Technology
Electrical Engineering Dept
Pittsburgh, Pa 15213

1 University of Mighigan
Electrical Engineering Dept
Ann Arbor, Michigan 48104

1 New York University
College of Engineering
New York, N.Y. 10019

1 Dept of Electrical Engineering
Texas Technological College
Lubbock, Texas 79409

1 IBM Technical Info. Retrieval Center
International Business Machines Corp.
Armonk, New York 10504

1 Commander
Test Command (TCDT-E)
Defense Atomic Support Agency
Sandia Base
Albuquerque, New Mexico 87115

1 Commanding General
U.S. Army Weapons Command
Rock Island, Ill 61201
Attn: ANSWE-RDR (Gerald Reinsmith)

1 Col E.P. Gaines, ACDA/FO
1901 Pennsylvania Ave, N.W.
Washington, D.C. 20451

1 Mr Billy Locke
Plans Directorate
USAF Security Service
Kelly Air Force Base, Texas 78241

1 W.A. Eberspacher
Technical Consultant
Systems Integration
Code 5340A, Box 15
Naval Missile Center
Point Magu, California 93041

1 Director of Faculty Research
Department of the Air Force
USAF Academy
Colorado 80840

1 Lt Col Richard Bennett
AFRDD
The Pentagon
Washington, D.C. 20301

1 Weapons Systems Evaluation Group
Attn: Col John B. McKinney
410 Army-Navy Drive
Arlington, Virginia 22202

1 Mr M. Zane Thornton
National Library of Medicine
8600 Rockville Pike
Bethesda, Maryland 22014

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

University of Illinois
Coordinated Science Laboratory
Urbana, Illinois 61801

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

ISL - A STRING MANIPULATING LANGUAGE

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

K. C. Kelley, S.R. Ray & F.A. Stahl

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February, 1969	27	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAB 07-67-C-0199	R-407
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this report is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program thru U.S. Army Electronics Command Ft. Monmouth, New Jersey 07703

13. ABSTRACT

In this paper the Information Search Language (ISL) is described. ISL is a problem-oriented language designed to facilitate the manipulation of real character strings.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Programming Language						
String Manipulation						
Character Manipulation						
Interactive Language						